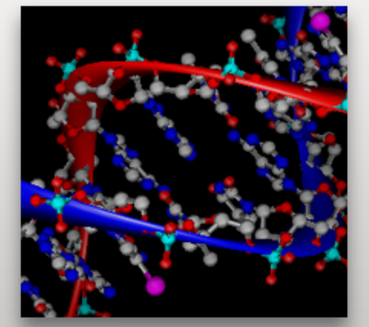
Smart Cards in Mac OS X

Session 502

These are confidential sessions—please refrain from filming or taking pictures.



OS Foundations

Smart Cards in Mac OS X

Michael Brouwer
Sr. Software Engineer, Data Security

http://developer.apple.com/wwdc2005/

What You'll Learn

- This session covers the use of smart cards in Mac OS X
- How to write applications that support smart cards
- How to add support for new types of cards

Progress

http://developer.apple.com/wwdc2005/

Demo

New smart card functionality

John Hurley
Data Security Group

http://developer.apple.com/wwdc2005/

Progress Since Mac OS X 10.3

- Login
- Email
  - Signing
  - Encryption (when suported)
- ssl client side auth
- VPN
- 802.1x

What Do Applications Need to Do?

As an application developer how do I get smart card support?

Demo

How to add smart card support

John Hurley
Data Security Group

http://developer.apple.com/wwdc2005/

Adding Smart Card Support

Entourage

- Existing Mac OS X application
- Now works with smart cards
- Without changes to the application itself

Integration

- How to get this level of integration in your apps?
- Use CDSA or CDSA based APIs
  - SecKeyRef's and friends
  - SecureTransport
  - CFNetwork
  - WebKit

Smart Cards in Mac OS X Tiger

http://developer.apple.com/wwdc2005/

Goals

- Zero user configuration
- Works with existing application using Mac OS X APIs
- Single sign-on

- We examined various options
  - Write a CSP/DL per card and load it in each app
    - Doesn’t meet the single sign-on goal
  - Write a PKCS11 driver for each card
    - Doesn’t meet the single sign-on nor existing API goals

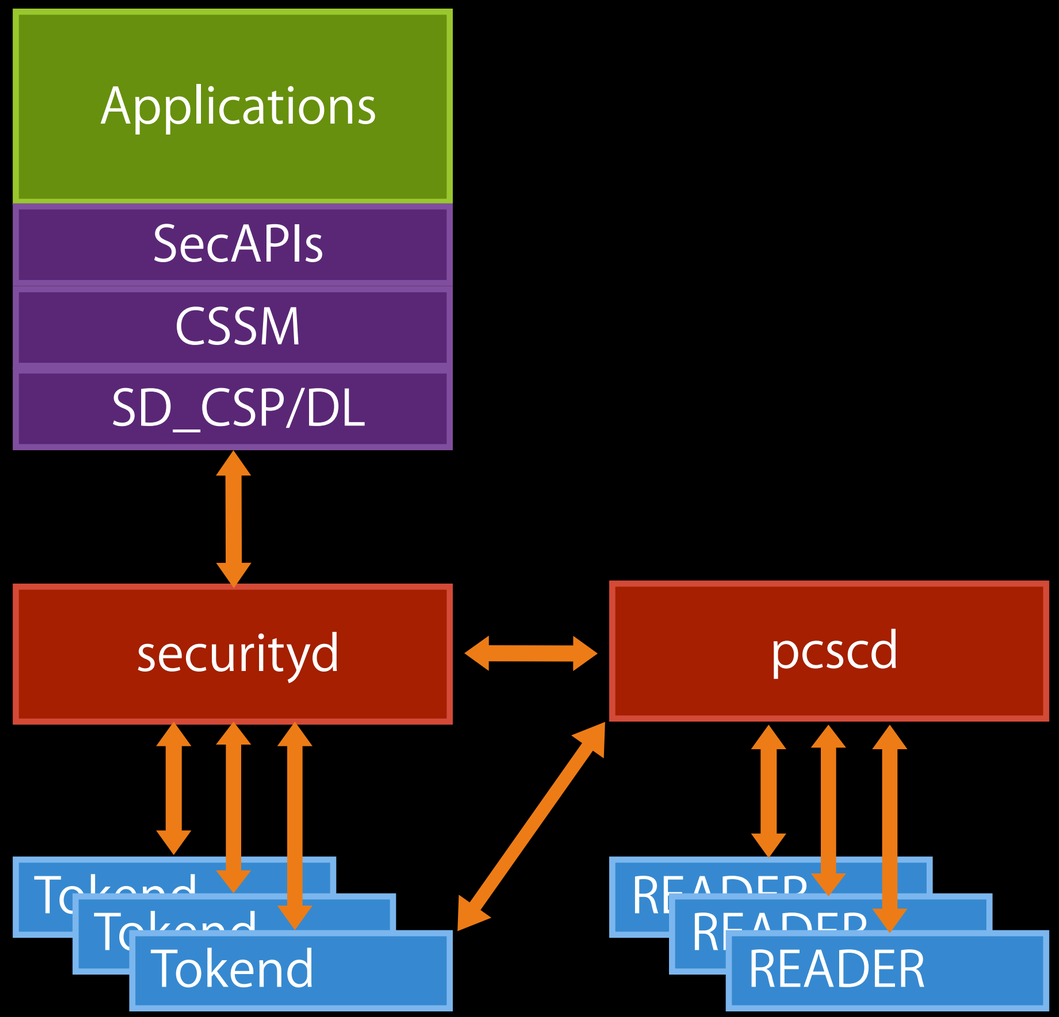
- Our solution…

Smartcard Support

- Horizontal smartcard support
  - Login
  - Authorization
  - Mail
  - Safari
  - Screen locking
  - Any apps using CDSA or CDSA based APIs
- Single sign-on with smartcard
- ACL support
- Previously only vertical smartcard solutions based on PCSC available

Smart Cards as Keychains

Architecture



Tokend

- What is Tokend?
  - Server proccess loaded for each token (or card) inserted
  - One tokend per active token
  - CDSA CSP/DL inspired SPI
  - Implements a number of callback functions which are invoked by the system as needed
- Each token is a keychain from the API point of view
- Tokens are automatically added to the new dynamic search list
- Searched right after the users file based keychains

Tokend

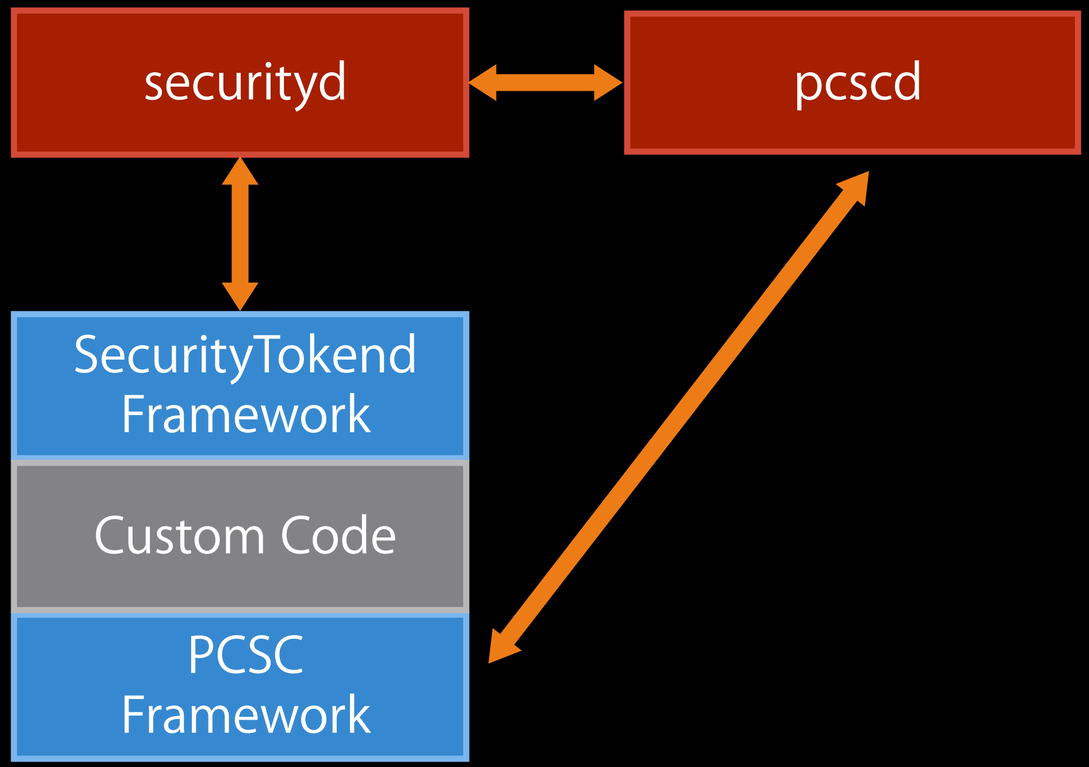
- What is Tokend?
  - Server proccess loaded for each token (or card) inserted
  - One tokend per active token
  - CDSA CSP/DL inspired SPI
  - Implements a number of callback functions which are invoked by the system as needed
- Each token is a keychain from the API point of view
- Tokens are automatically added to the new dynamic search list
- Searched right after the users file based keychains

How to Write Your Own Tokend

- Implement the callback interface in SecurityTokend's SecTokend.h header
- C callback interface
- We use a C++ layer on top of the C interface to implement our tokends
  - You can use either
  - The C++ layer helps make your tokend implement all the features needed for it to look like a normal keychain

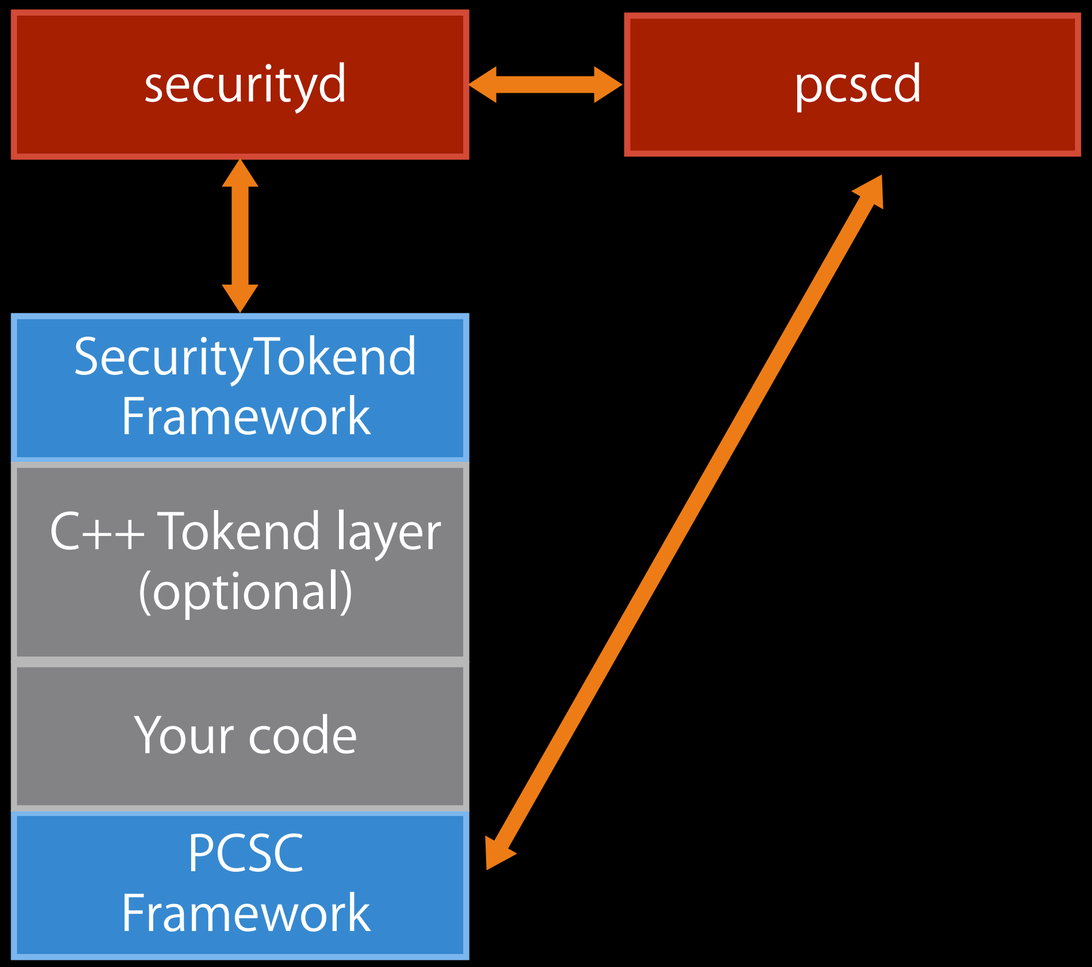
Tokend

Architecture



Tokend with C++ Layer

Architecture



Main Body of a Tokend

- Main part of your tokend

```
SecTokendCallbacks callbacks = { ... };
SecTokendSupport support;
int main(int argc, char **argv) {
return SecTokendMain(argc, argv, &callbacks, &support);
}
```

- `callbacks` contains function pointers to your tokend
- `support` will contain callbacks for allocating data to be returned to the caller. Most importantly a way to get the `SCARD_READERSTATE` which describes the pcsc slot this tokend is servicing

Implementing Probe

```
CSSM_RETURN probe(SecTokendProbeFlags flags, uint32 *score,
char tokenUid[TOKEND_MAX_UID]) {
}
```

- Probe should return a nonzero score if the token in the slot identified by the `SCARD_READERSTATE *` returned by calling `support.startupReaderInfo();` is supported by this tokend
- If a nonzero score is returned probe should populate `tokenUid` with a unique id for this token if possible

You're It: Establish

- Once the system has decided your tokend has the highest score of all available tokend's for a given token establish is called

```
CSSM_RETURN establish(const CSSM_GUID *guid, uint32 subserviceId,
SecTokendEstablishFlags flags, const char *cacheDirectory, const char
*workDirectory, char mdsDirectory[PATH_MAX], char printName[PATH_MAX]) {
}
```

Object Types

- Tokens can contain the following object types
  - X509 Certificates
  - Keys
    - Public
    - Private
    - Symmetric
  - Generic
  - Some custom type
    - You should probably use generic in this case
- A PIN is not a first class object—more on this later

Data and Attribute Retrieval Calls

- There are 3 ways a key or object's data and attributes can be obtained:
  - `findFirst`
  - `findNext`
  - `findRecordHandle`
- Attributes and data are returned using the allocators in the support callbacks
- Certificates and key must have very specific attributes for the higher levels to see them as keychain items
  - This is where the C++ layer helps simplify things a lot

findFirst

```
CSSM_RETURN findFirst(const CSSM_QUERY *query, TOKEND_RETURN_DATA
*data, CSSM_HANDLE *hSearch) {
}
```

- Queries can be quite complex, higher level APIs rely on most query features being there
- The C++ layer fully supports all CSSM style queries

TOKEND_RETURN_DATA

```
typedef struct {
  CSSM_DB_RECORD_ATTRIBUTE_DATA *attributes; // add attribute values here
  CSSM_DATA *data;                 // store item data there unless NULL
  CSSM_HANDLE record;              // store record-id here
  CSSM_HANDLE keyhandle;           // key handle if key, 0 otherwise
} TOKEND_RETURN_DATA;
```

- Both attributes and data are optional. Keys and object have separate handles and may be released independently.

findNext

```
CSSM_RETURN findNext(CSSM_HANDLE hSearch, TOKEND_RETURN_DATA *data) {
}
```

- Return the next matching object or key in an already started query

findRecordHandle

```
CSSM_RETURN findRecordHandle(CSSM_HANDLE hRecord,
TOKEND_RETURN_DATA *data) {
}
```

- Retrieve attributes or data from an object handle

Access Control Lists

http://developer.apple.com/wwdc2005/

ACL Calls

- get*Owner
- get*Acl
- change*Owner
- change*Acl
- Where * is Database, Object or Key

ACL Components

ACLs consist of one or more entries, each of which contain:

- Tag (optional label for an entry)
- Authorization(s) (operations this acl entry allows)
  - db_read, write, decrypt, sign
- Subject (what is needed to authorize this acl entry)
  - protected_password, threshold, preauth
  - Some subjects recurse and contain additional acl entries
  - Some subjects contain one or more values
    - May be public or private
    - Matched against Sample of a Credential during evaluation
    - Private values not returned by get*acl calls

Entities Which Have ACLs

- Database
- Key
- Object (unique to tokend)

PIN ACLs

- A PIN ACL will typically have a single entry with subject type `CSSM_ACL_SUBJECT_TYPE_PREAUTH_SOURCE`
- Other ACLs may refer to the PIN by using a `CSSM_ACL_SUBJECT_TYPE_PREAUTH` type subject
- The tag of a PIN ACL must be PIN# (where # is a number)
  - Some applications assume PIN1 is the default so if you have a "default" pin call it PIN1
- PINs cannot refer to each other, and PINs are not special
- A PIN ACL will just allow or disallow a particular operation

ACL Tips

- Before each operation that uses an object from the token the db_read authorization on the database ACL is checked
  - Unless you require a PIN to read anything from the card this ACL should have subject `CSSM_ACL_SUBJECT_TYPE_ANY`
- Subsequently the object or key ACL is checked for the operation
  - The find* calls require db_read authorization on the item
  - Make sure keys are always readable
    - This does not imply extractable which is a separate authorization

ACL Wrap Up

- Tokend is responsible for enforcing the ACLs on databases, keys and objects (in practice the token itself will often do this)
- The system (securityd) will use the ACLs to preflight UI and acquire the appropriate PIN(s) from the user

Crypto Operations

http://developer.apple.com/wwdc2005/

Crypto Operations on Keys

- decrypt
- generateSignature
- unwrapKey

- Similar to normal CSP operations of the same names

- wrapKey, encrypt and verify are not required to be implemented for public keys since the higher levels will extract the key and use the built in CSP to do public key operations
- If you support symmetric keys you might need to implement them

decrypt

```
CSSM_RETURN decrypt(const CSSM_CONTEXT *context, CSSM_HANDLE hKey,
const CSSM_DATA *cipher, CSSM_DATA *clear) {
}
```

- hKey must have been obtained though one of the find* calls
- context contains optional additional information like initvectors, padding modes, etc.

generateSignature

```
CSSM_RETURN generateSignature(const CSSM_CONTEXT *context, CSSM_HANDLE
hKey, CSSM_ALGORITHMS signOnly, const CSSM_DATA *input, CSSM_DATA
*signature) {
}
```

- No surprises here, very similar to decrypt

unwrapKey

```
CSSM_RETURN unwrapKey(const CSSM_CONTEXT *context, CSSM_HANDLE hWrappingKey,
const CSSM_KEY *wrappingKey, const CSSM_ACCESS_CREDENTIALS *cred, const
CSSM_ACL_ENTRY_PROTOTYPE *access, CSSM_HANDLE hPublicKey, const CSSM_KEY
*publicKey, const CSSM_KEY *wrappedKey, CSSM_KEYUSE usage,
CSSM_KEYATTR_FLAGS attributes, CSSM_DATA *descriptiveData, CSSM_HANDLE
*hUnwrappedKey, CSSM_KEY *unwrappedKey) {
}
```

- Needed for S/MIME decryption: unwrapping a symmetric key that was wrapped by a public key
- Other operations can be supported if needed

Tokend Summary

- Allows shared access to smart cards or other tokens
- Fairly simple to implement, especially with the C++ layer
- Tokend is not PKCS#11
- It's possible to implement a tokend on top of most PKCS#11 libraries except for the probe call
- In most cases writing a tokend is less work than writing a PKCS#11 library
- We didn't talk about modifying the token
  - The callbacks to do so are there

Existing Smartcard Solutions

- Existing vertical smartcard solutions should continue to work
- The tokend's we ship don't grab exclusive access to any cards, they use transactions

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

Security Feedback Forum	Session 511 Friday 5:00PM

Who to Contact

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Q&A

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference